NATIONAL UNIVERSITY OF SINGAPORE

Department of Electrical and Computer Engineering

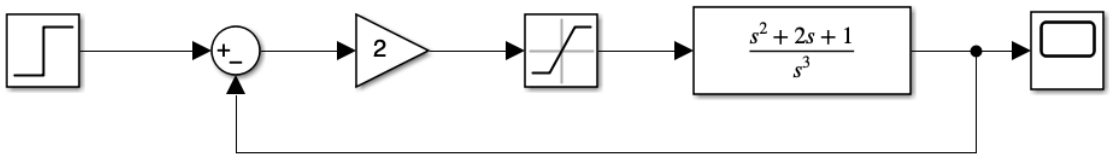
EE4302 Advanced Control Systems Experiment II
Nonlinear System

E0260014Y
Guo Shiping

Table of Contents

Table of figures

1. Root-locus analysis



1.1 Simulink plots

Input = 1

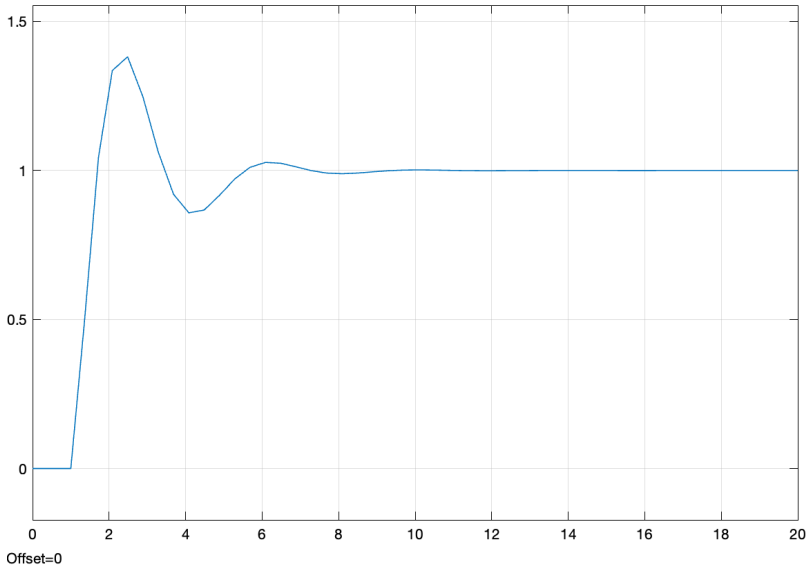


Figure 1 Step response with input is 1

Input = 2

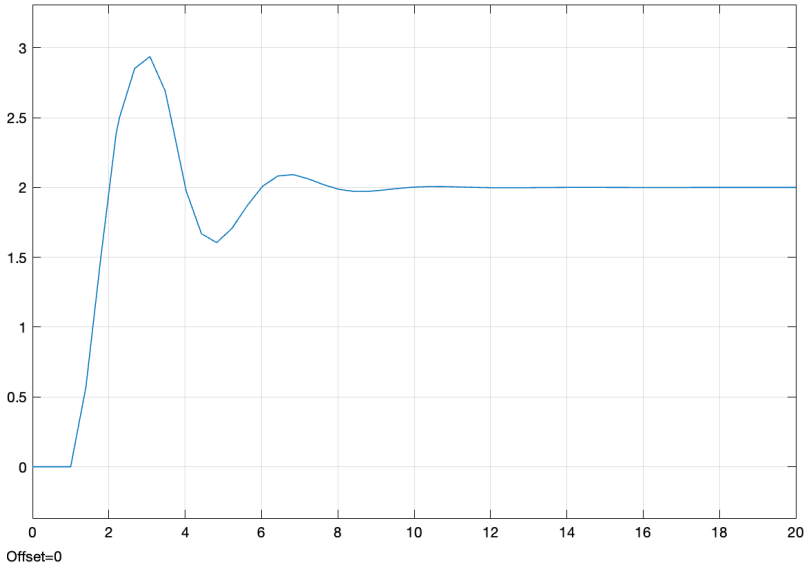


Figure 2 Step response with input is 2

Input = 3

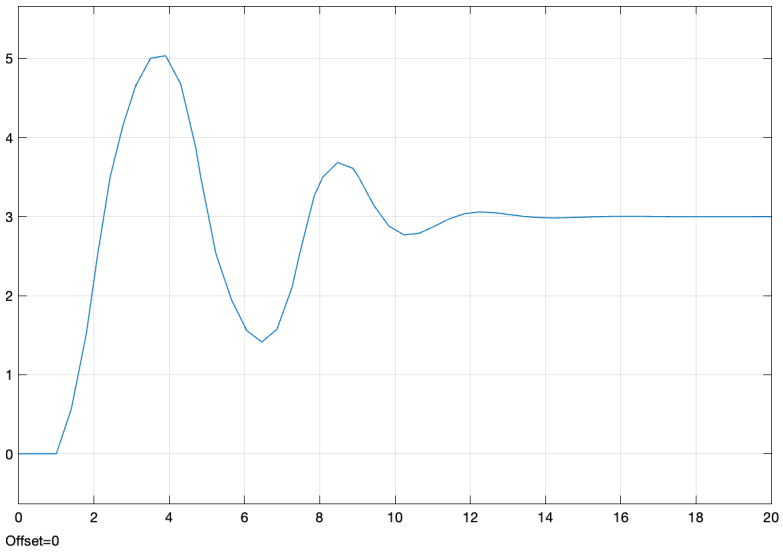


Figure 3 Step response with input is 3

Input = 4

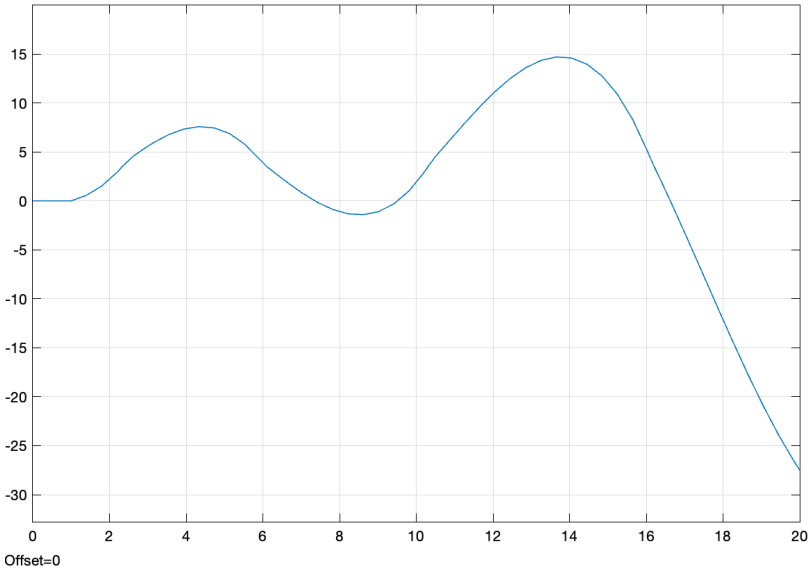


Figure 4 Step response with input is 4

Note down the period of the sustained oscillation.

Input x	Period of the sustained oscillation
1	8
2	10
3	14
4	Unstable

The behaviour of the system can be qualitatively described by considering the Saturation block as a varying signal-dependent gain.

1.2 Root-locus analysis

The closed loop transfer function is:

$$\frac{K(s+1)^2}{s^3+K(s+1)^2}$$

The poles are:

$$s^3+K(s+1)^2=0$$

This three order polynomials can't be solved by hand

The solutions from MATLAB are:

```
poles =

          -0.3487
0.1244 + 0.5208i
0.1244 - 0.5208i
```

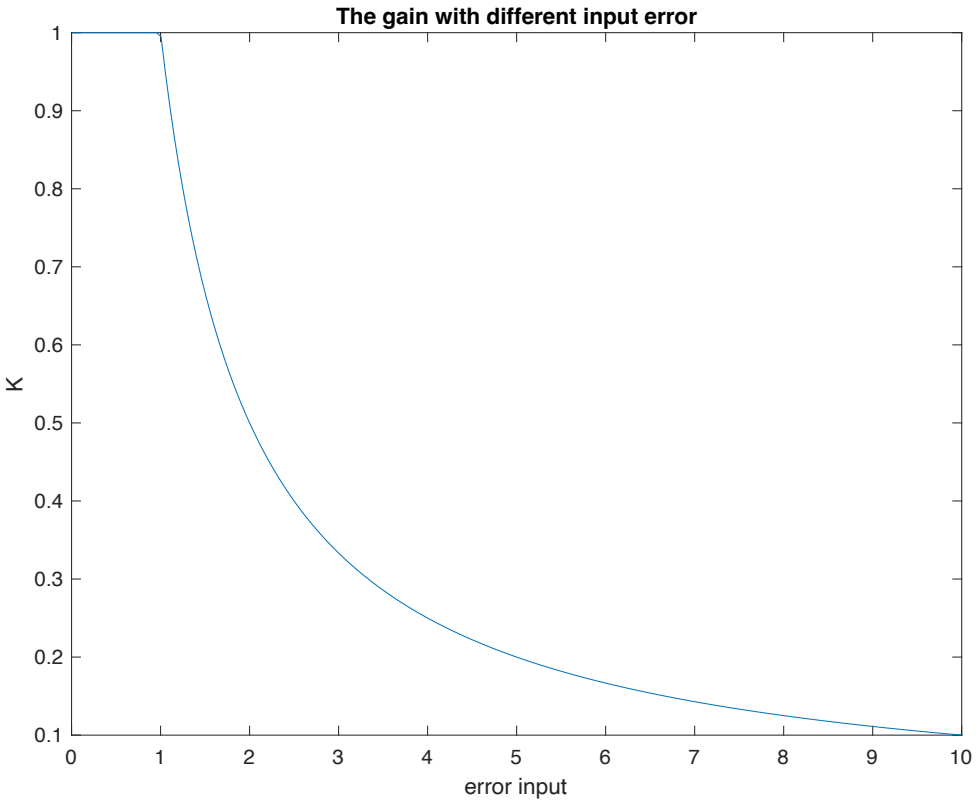


Figure 5 K changes with different input size

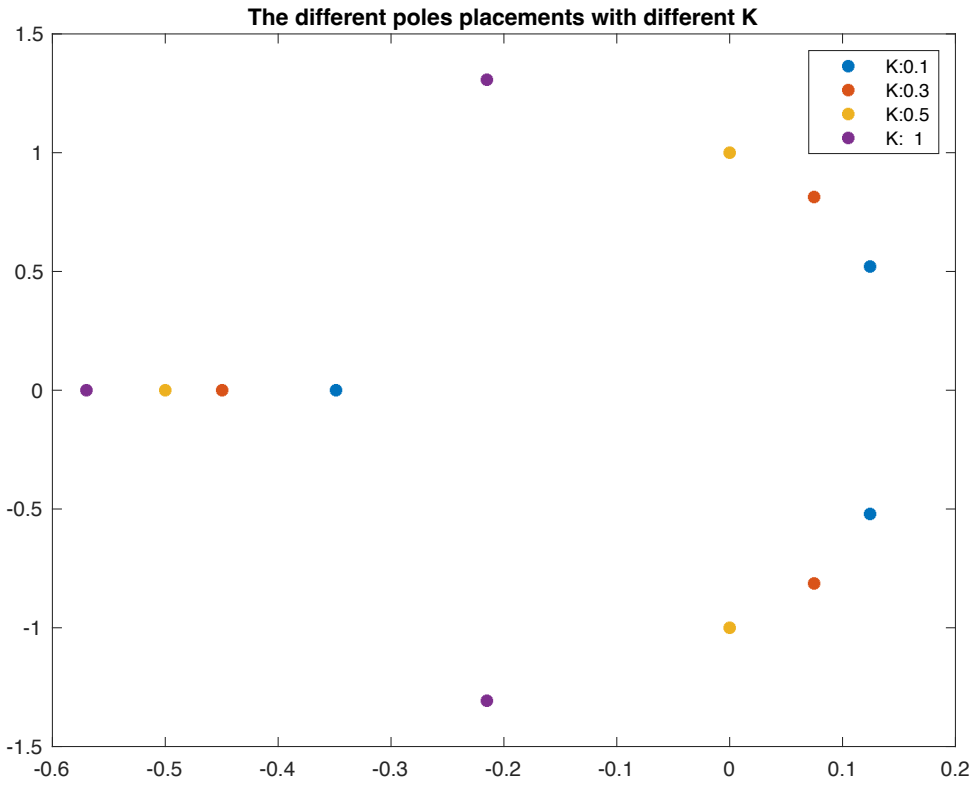


Figure 6 Poles changes with different K

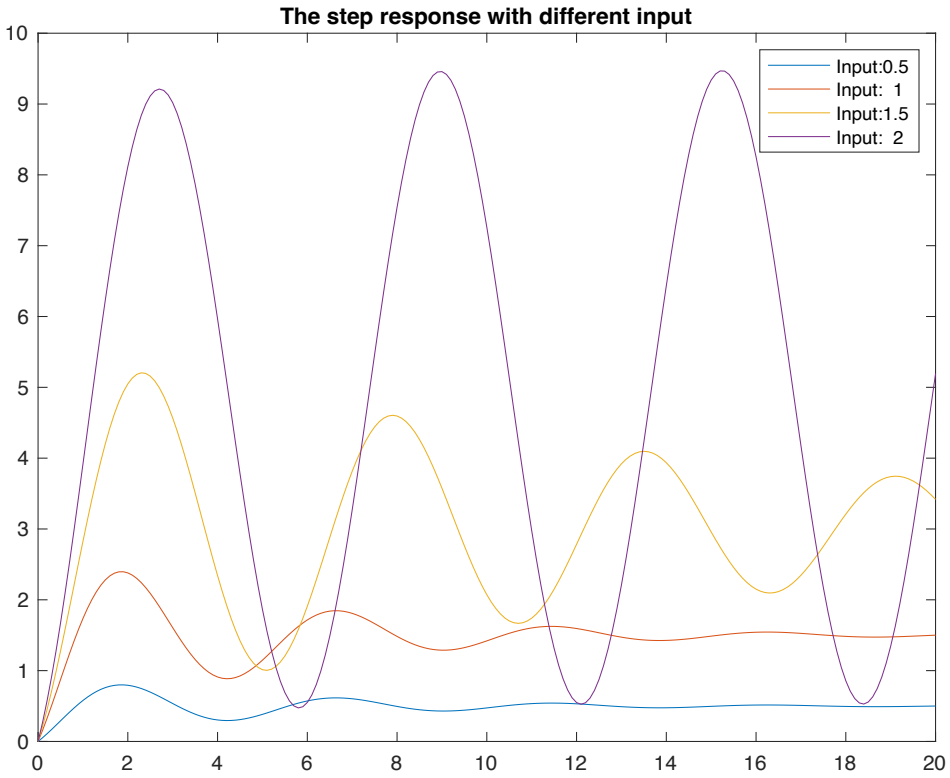


Figure 7 Step response with different size input

We can see when the input size is 2, the response settling time is very long, the result is near to blow up. The result is the same as the Simulink showed above.

1.3 Analyse results

The result is very clear, it is the same as what we conclude in class.

The outcome aligns precisely with our class discussions. In this system, a larger input leads to a smaller value of K. Consequently, with a smaller K, the poles depicted in Figure 6 exhibit a greater distance from their original positions. As a result, ω_n is smaller. The angle with the y-axis is also reduced, resulting in a smaller ς . This contributes to shorter settling times and diminished overshoot compared to the scenario with a larger K. Figure 7 distinctly illustrates this finding, showcasing that smaller K values correspond to minimal overshoot and quicker settling speeds.

This outcome contrasts sharply with linear systems, where identical overshoot and settling times are typically observed for the same system. The non-linear system, as evidenced by our analysis, exhibits a notable departure from this linear behaviour.

2. Sliding mode control design

Consider the unstable system

$$\begin{bmatrix} \dot{x_1} \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u$$

which has the transfer function

$$\frac{Y(s)}{U(s)} = \frac{1}{s(s-1)}$$

(a) Design a variable-structure controller. Choose the switching line as

$$\sigma = x_1 + x_2$$

and the amplitude of the “sign" function as 0.5.

(b) Simulate the system using Simulink. Plot the states, control signal and phase plane trajectory.

(c) Modify the variable-structure controller such that chattering is eliminated and simulate the system using Simulink. Plot the states and control signal.

(d) Analyse your results.

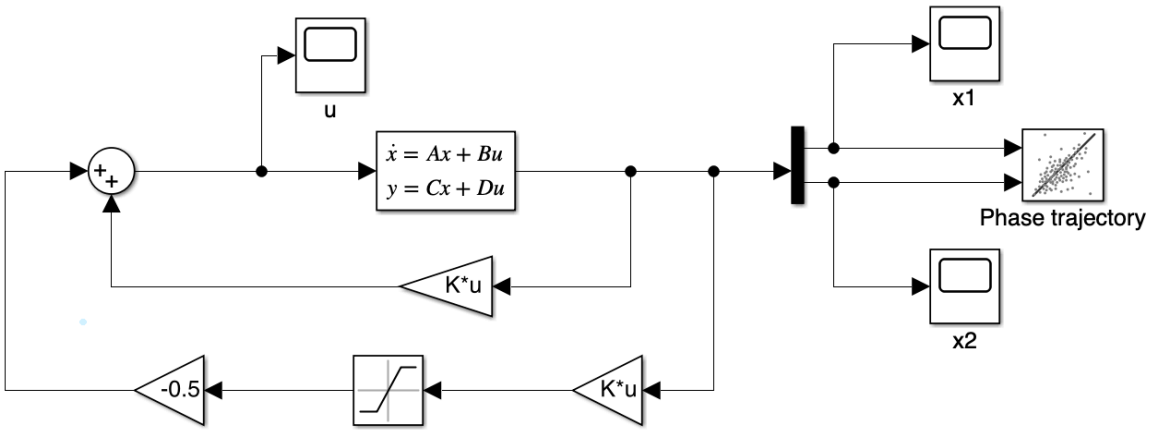


Figure 8 Sliding mode control Simulink design

2.1 Simulink plots

x1

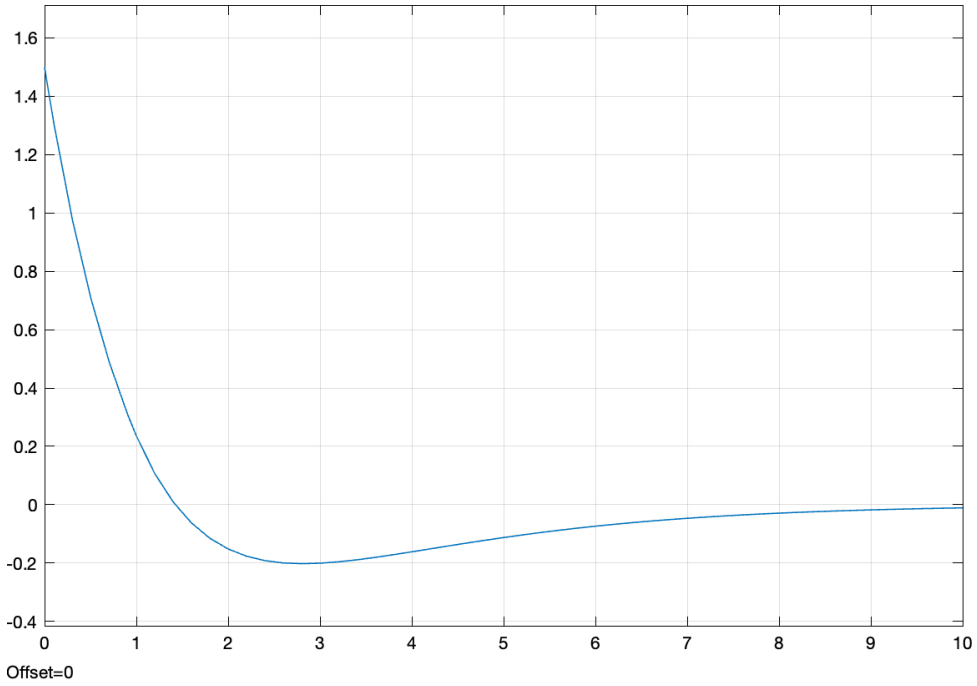


Figure 9 Sliding mode control x1

x2

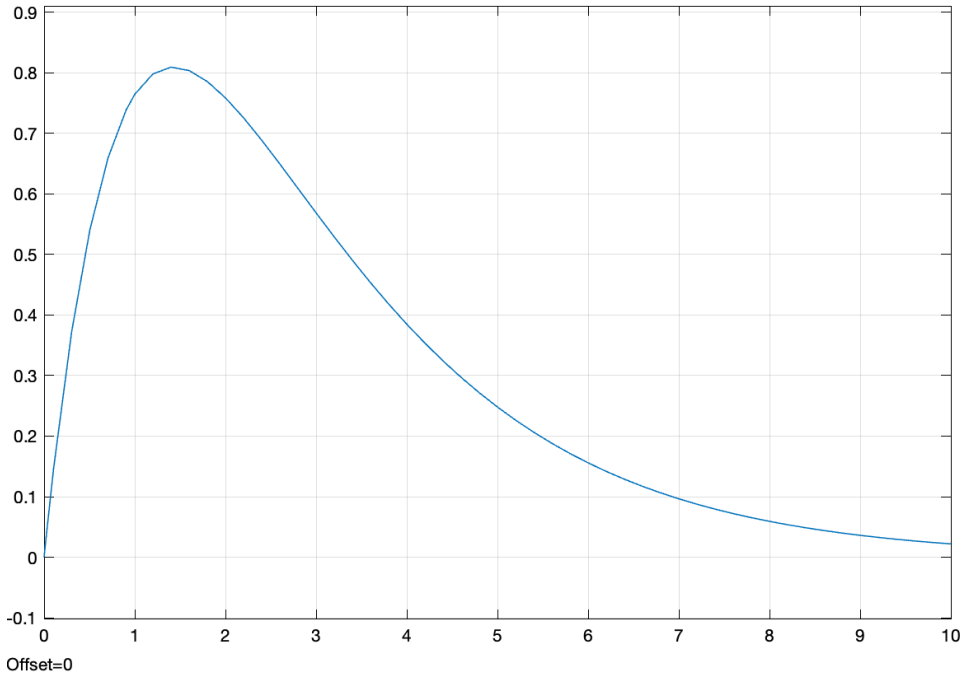


Figure 10 Sliding mode control x2

Input u

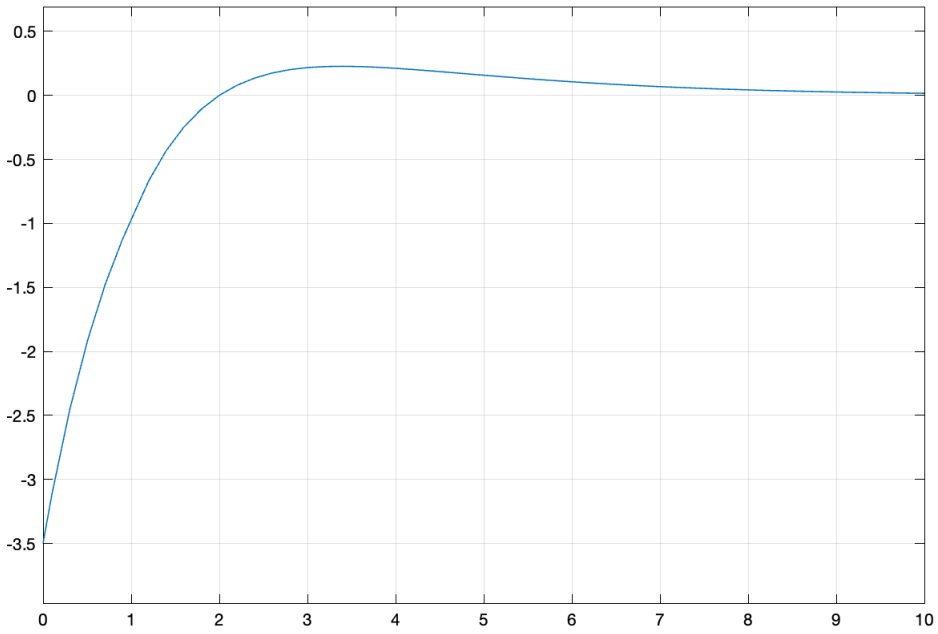


Figure 11 Sliding mode control input u

Phase plan trajectory

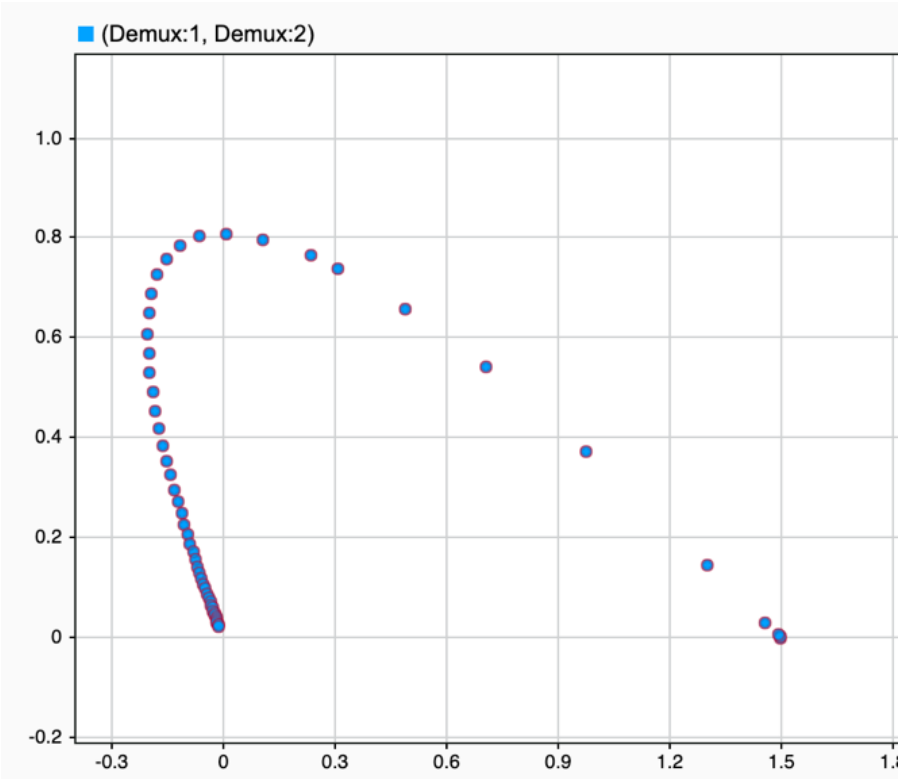


Figure 12 Phase trajectory

Use Sign function the control signal is：

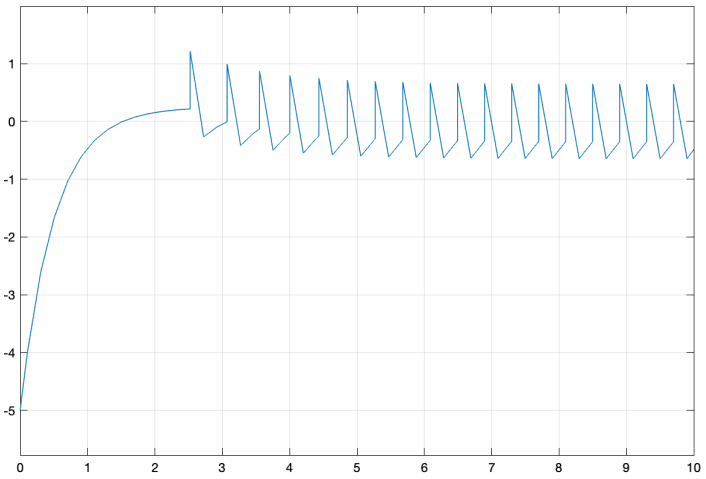


Figure 13 Control signal when using sign function

We can see the chattering problem is serious.

Use Saturation function:

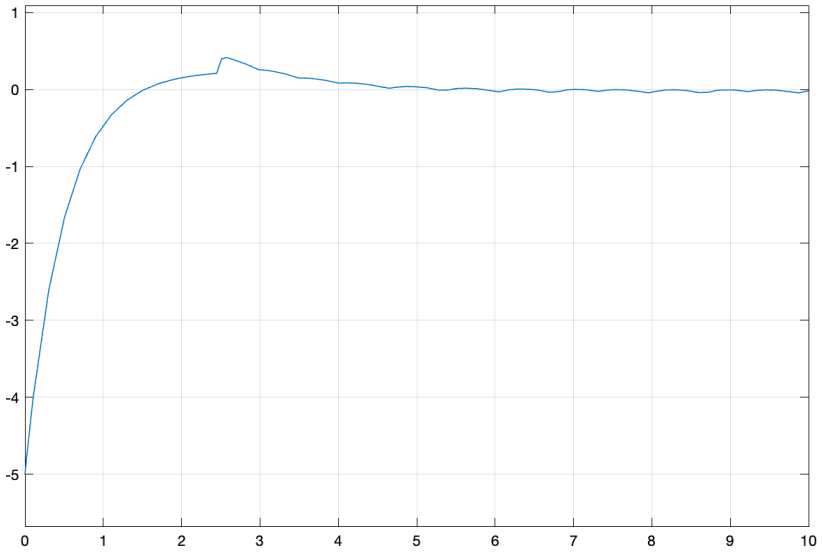


Figure 14 Control signal when using saturation function

The chattering problem is relieved a lot.

2.2 Further analysis

Setting the initial state in different number and see the difference.

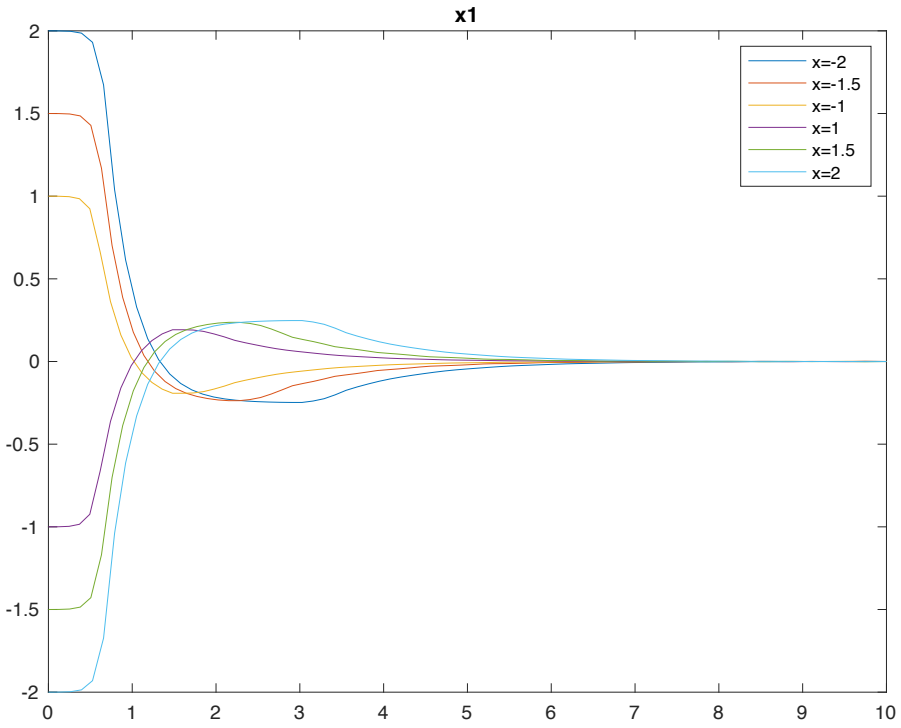


Figure 15 x1 states under different initial states

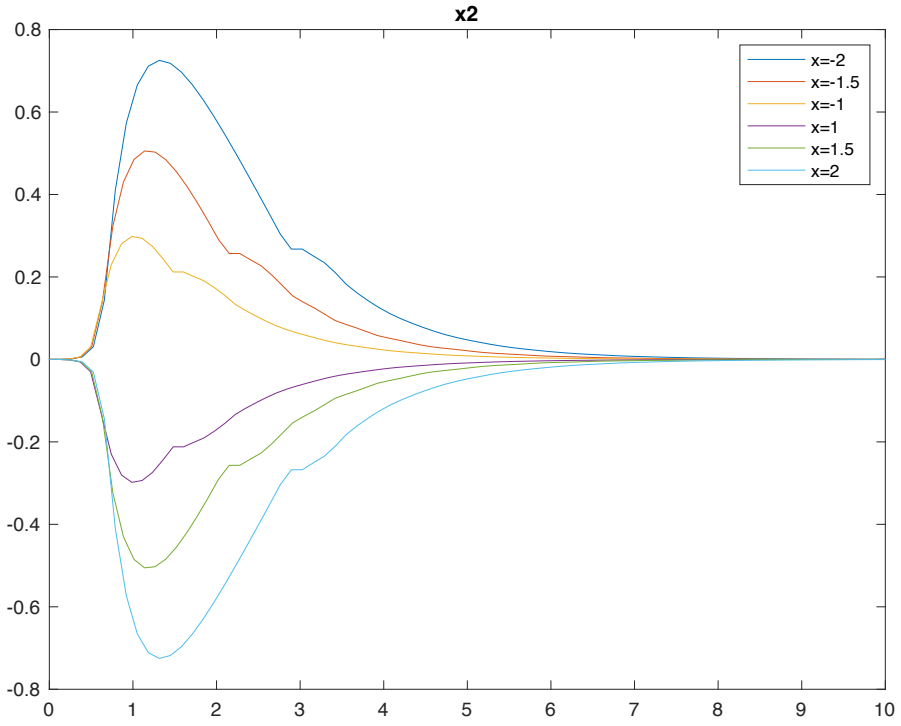


Figure 16 x2 states under different initial states

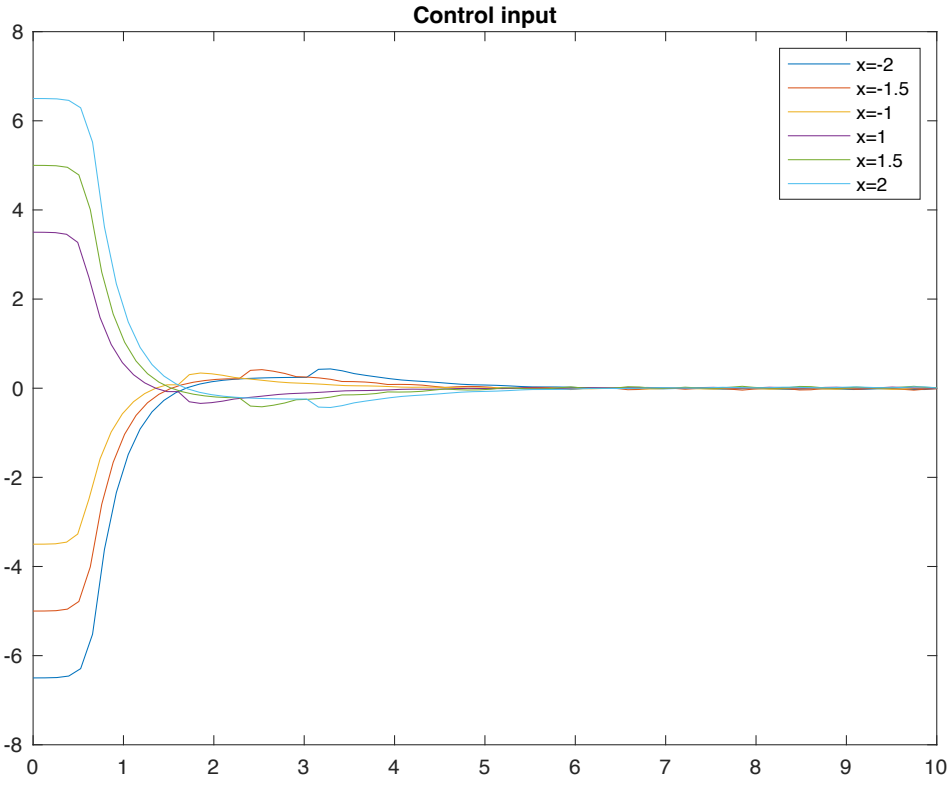


Figure 17 Control signal under different states

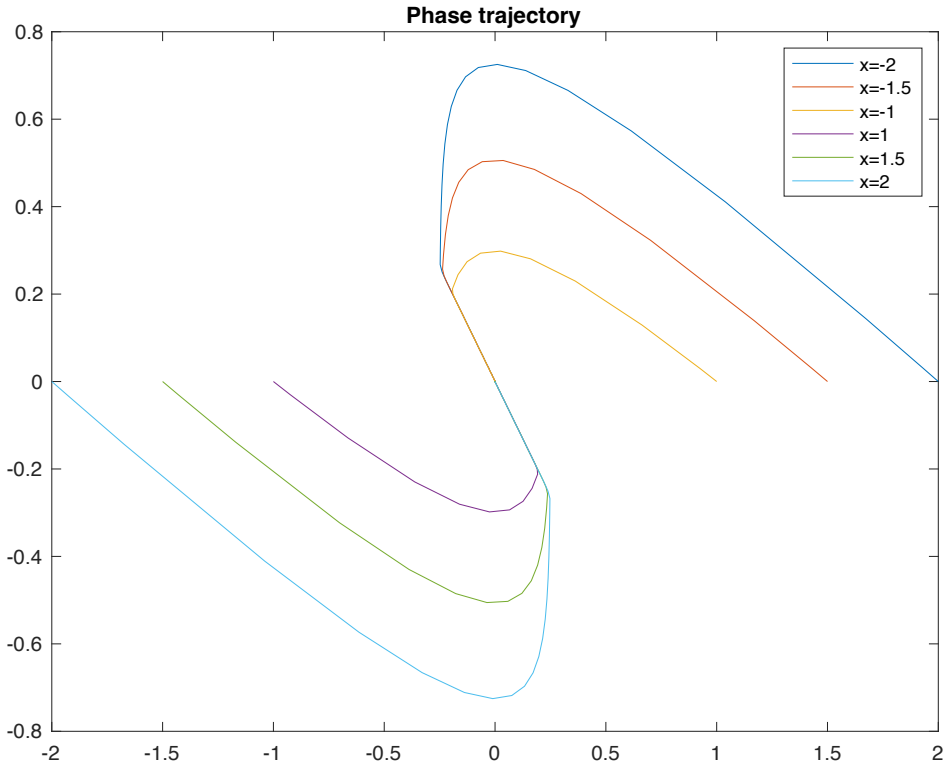


Figure 18 Phase trajector under different states

2.3 Discussion

Figures 15-18 demonstrate rapid convergence of states and control signals under different initial conditions, affirming the effectiveness of the control method.

I recently acquired knowledge of LQR control in another course, and from my perspective, the derivative process appears quite similar to the LQR method. Both methods seem to be optimal approaches, employing analogous strategies to achieve their objectives. The primary distinction lies in the utilization of the Sliding Control method for non-linear systems. This imparts robustness to uncertainties and disturbances, as the control law is designed to remain effective even when precise knowledge of system parameters or external disturbances is lacking.

The application of sliding control was evident in its ability to stabilize an initially unstable system. The states converged to a desired equilibrium point, highlighting the efficacy of the sliding mode control approach.

Furthermore, the chattering problem, a common issue, can be effectively addressed by introducing a saturation function. This function proves highly adept at resolving the chattering problem, and by adjusting its characteristics, we can modify and enhance the results.

3. Conclusion

I have limited knowledge about non-linear systems despite having heard about them for quite some time. The only thing I'm aware of is that they adhere to the superposition principle. This assignment is valuable for gaining a broad understanding of non-linear systems and learning how to control them using sliding control.

In the initial section, I comprehend the characteristics of non-linear systems, recognizing their stark differences from linear systems. Intuition and linear logic don't apply in the same way, and I'm familiar with how input impacts the entire system, employing the root locus analysis method for system analysis.

In the latter part, grasping derivatives proves challenging, yet it's instrumental in comprehending how to control a non-linear system, which fundamentally differs from its linear counterpart. Despite the distinctions, the overarching goal remains consistent—achieving system stability through mathematical approaches.

Appendix:

```
%% Root locus analysis
clc
clear
close all

syms s z
K = [0.1, 0.3, 0.5, 1];
for i = 1 : length(K)
    k = K(i);
    equ = s^3+k*(s+1)^2== 0;
    poles = vpa(solve(equ, s), 4);
    pole1_r = real(poles(1));
    pole1_i = imag(poles(1));
    pole2_r = real(poles(2));
    pole2_i = imag(poles(2));
    pole3_r = real(poles(3));
    pole3_i = imag(poles(3));
    x = [pole1_r, pole2_r, pole3_r];
    y = [pole1_i, pole2_i, pole3_i];
    plot(x,y, '*', 'Linewidth', 2)
    hold on
end
legend({[ 'K:'; 'K:'; 'K:'; 'K:'] ,num2str(K' )]});
%legend(num2str(K))
title('The different poles placements with different K')

%% gain
% K = 1
% when error input bigger than -1 and smaller than 1 the feedback transfer
% function is 1/s^2, or it will be 1/s
num = [1 2 1];
den = [1 1 2 1];
sys1 = tf(num,den);
DC_gain = dcgain(sys1);
t = linspace(0, 10, 200);
y = zeros(1,length(t));
y(1, 1:20) = DC_gain;
y(1, 21:200) = 1./t(21:200);
plot(t,y)
xlabel('error input')
ylabel('K')
title('The gain with different input error')

%% different input response

t = 0:0.1:20;  % 201 points
u = zeros(1, length(t));
I = [0.5 1 1.5 2];
for i = 1 : length(I)
    if I(i) <=1
        k = 1;
    else
        k = 1/I(i);
    end
    num = [k 2*k k];
    den = [1 k 2*k k];
```

```
    sys = tf(num,den);
    u = I(i) + u;
    y = lsim(sys,u,t);

    plot(t, y)
    hold on
end
legend({[[ 'Input:'; 'Input:'; 'Input:'; 'Input:'] ,num2str(I' )]});
%legend(num2str(K))
title('The step response with different input')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% SLIDING MODE CONTROL DESIGN
% gTruth0=evalin('base', 'out_2.mat');
% signal0 = gTruth0.Phase.signals;
% data0_1 = signal0(1).values;
% data0_2 = signal0(2).values;
% ctr_input0 = signal0(3).values;
% save('out_2')
%
% gTruth1=evalin('base', 'out_1_5');
% signal1 = gTruth1.Phase.signals;
% data1_1 = signal1(1).values;
% data1_2 = signal1(2).values;
% ctr_input1 = signal1(3).values;
% save('out_1_5')
%
% gTruth2=evalin('base', 'out_1');
% signal2 = gTruth2.Phase.signals;
% data2_1 = signal2(1).values;
% data2_2 = signal2(2).values;
% ctr_input2= signal2(3).values;
% save('out_1')
%
% gTruth3=evalin('base', 'out_min1');
% signal3 = gTruth3.Phase.signals;
% data3_1 = signal3(1).values;
% data3_2 = signal3(2).values;
% ctr_input3 = signal3(3).values;
% save('out_min1')
%
% gTruth4=evalin('base', 'out_min1_5');
% signal4 = gTruth4.Phase.signals;
% data4_1 = signal4(1).values;
% data4_2 = signal4(2).values;
% ctr_input4 = signal4(3).values;
% save('out_min1_5')
%
% gTruth5=evalin('base', 'out_min2');
% signal5 = gTruth5.Phase.signals;
% data5_1 = signal5(1).values;
% data5_2 = signal5(2).values;
% ctr_input5 = signal5(3).values;
% save('out_min2')

load('out_2.mat')
```

```
N = length(data0_2);
t = linspace(0,10,N);

figure(1)
plot(linspace(0,10,length(data0_1)), data0_1)
hold on
plot(linspace(0,10,length(data1_1)), data1_1)
plot(linspace(0,10,length(data2_1)), data2_1)
plot(linspace(0,10,length(data3_1)), data3_1)
plot(linspace(0,10,length(data4_1)), data4_1)
plot(linspace(0,10,length(data5_1)), data5_1)
legend('x=-2', 'x=-1.5', 'x=-1', 'x=1', 'x=1.5', 'x=2')
title('x1')
hold off

figure(2)
plot(linspace(0,10,length(data0_2)), data0_2)
hold on
plot(linspace(0,10,length(data1_2)), data1_2)
plot(linspace(0,10,length(data2_2)), data2_2)
plot(linspace(0,10,length(data3_2)), data3_2)
plot(linspace(0,10,length(data4_2)), data4_2)
plot(linspace(0,10,length(data5_2)), data5_2)
legend('x=-2', 'x=-1.5', 'x=-1', 'x=1', 'x=1.5', 'x=2')
hold off
title('x2')

figure(3)
plot(data0_1, data0_2)
hold on
plot(data1_1, data1_2)
plot(data2_1, data2_2)
plot(data3_1, data3_2)
plot(data4_1, data4_2)
plot(data5_1, data5_2)
legend('x=-2', 'x=-1.5', 'x=-1', 'x=1', 'x=1.5', 'x=2')
hold off
title('Phase trajectory')

figure(4)
plot(linspace(0,10,length(ctr_input0)), ctr_input0)
hold on
plot(linspace(0,10,length(ctr_input1)), ctr_input1)
plot(linspace(0,10,length(ctr_input2)), ctr_input2)
plot(linspace(0,10,length(ctr_input3)), ctr_input3)
plot(linspace(0,10,length(ctr_input4)), ctr_input4)
plot(linspace(0,10,length(ctr_input5)), ctr_input5)
legend('x=-2', 'x=-1.5', 'x=-1', 'x=1', 'x=1.5', 'x=2')
hold off
title('Control input')
```